Новые возможности Tarantool 2.0

Никита Петтик
Кирилл Юхин

Новые возможности 2.0

- Core SQL support
  - Console
  - Strict typing
  - Interoperability with Lua
- New app server features: SQL constraints, triggers

Цели релиза 2.0

Главная цель - повысить доступность нашей in-memory технологии

- Гибкие возможности работы с данными
- Интеграция с экосистемой через ODBC: Wordpress, clikview etc.
- Сценарии миграции с классических СУБД: zero-tuning site
- Consistency в широком смысле. Strict typing, referential integrity, check constraints. Для энтерпрайза

Экосистема

Joomla!®
...because open source matters

ODBC

Perl DBI

Работа с данными

```
CREATE TABLE t1 (id INTEGER PRIMARY KEY, a INTEGER, b INTEGER, c INTEGER)
CREATE TABLE t2 (id INTEGER PRIMARY KEY, x INTEGER, y INTEGER, z INTEGER)
```

```
function query()
    local join = {}
    for _, v1 in box.space.t1:pairs({}, {iterator='ALL'}) do
        local v2 = box.space.t2:get(v1[1])
        if v2[3] > 1 then
            table.insert(join, {t1=v1, t2=v2})
        end
    end
    local dist = {}
    for _, v in pairs(join) do
        if dist[v['t1'][2]] == nil then
            dist[v['t1'][2]] = 1
        end
    end
    local result = {}
    for k, _ in pairs(dist) do
        table.insert(result, k)
    end
    return result
end
query()
```

```
box.sql.execute([[select distinct(a)
                  from t1, t2
                  where t1.id=t2.id and
                        t2.y > 1;]])
```

Императивы

- Interoperability: ANSI
- In-memory: ultra fast query processing

Tarantool SQL

- ACID transactions, SAVEPOINTs
- left/inner/natural JOIN, UNION/EXCEPT, subqueries
- HAVING, GROUP BY, ORDER BY
- WITH RECURSIVE
- Triggers
- VIEWs
- FOREIGN KEYS
- COLLATIONs

Триггеры

- Персистентны
- INSTEAD OF для VIEW
- Только FOR EACH ROW

```
CREATE TRIGGER after_insert AFTER INSERT ON t1
BEGIN
   DELETE FROM t1 WHERE x = (SELECT min(y) FROM t3);
   INSERT INTO t2(id, z) VALUES (NEW.id, 'ins');
END;
```

CHECK

- Interoperable: можно вызвать из Lua
- Произвольные SQL выражения

```
tarantool> opts = {checks = {{expr = 'X>5'}}}
tarantool> t = box.schema.space.create{'t', 'memtx', 0, opts, format}
tarantool> box.space._space:insert(t)

CREATE TABLE t(id PRIMARY KEY CHECK (id > 0));
```

Простой запрос

```
CREATE TABLE t1(n INTEGER PRIMARY KEY, log INTEGER);

tarantool> box.sql.execute([[SELECT log AS x, count(*) AS y
                             FROM t1
                             GROUP BY x
                             HAVING y >= 4
                             ORDER BY max(n)+1]]);

- - -

- - [3, 4]

  - [4, 8]

  - [5, 15]

...
```

Вложенный запрос

```
CREATE TABLE t1(n INTEGER PRIMARY KEY, log INTEGER);

tarantool> box.sql.execute([[SELECT a.y, a.count(*), max(x), count(*)
                             FROM
                               (SELECT count(*),y FROM t1 GROUP BY y) AS a,
                               (SELECT max(x),y FROM t1 GROUP BY y) as b
                             WHERE a.y=b.y ORDER BY a.y]])

---
- - [1, 1, 1, 1]
  - [2, 2, 3, 2]
  - [3, 2, 5, 2]
...
```

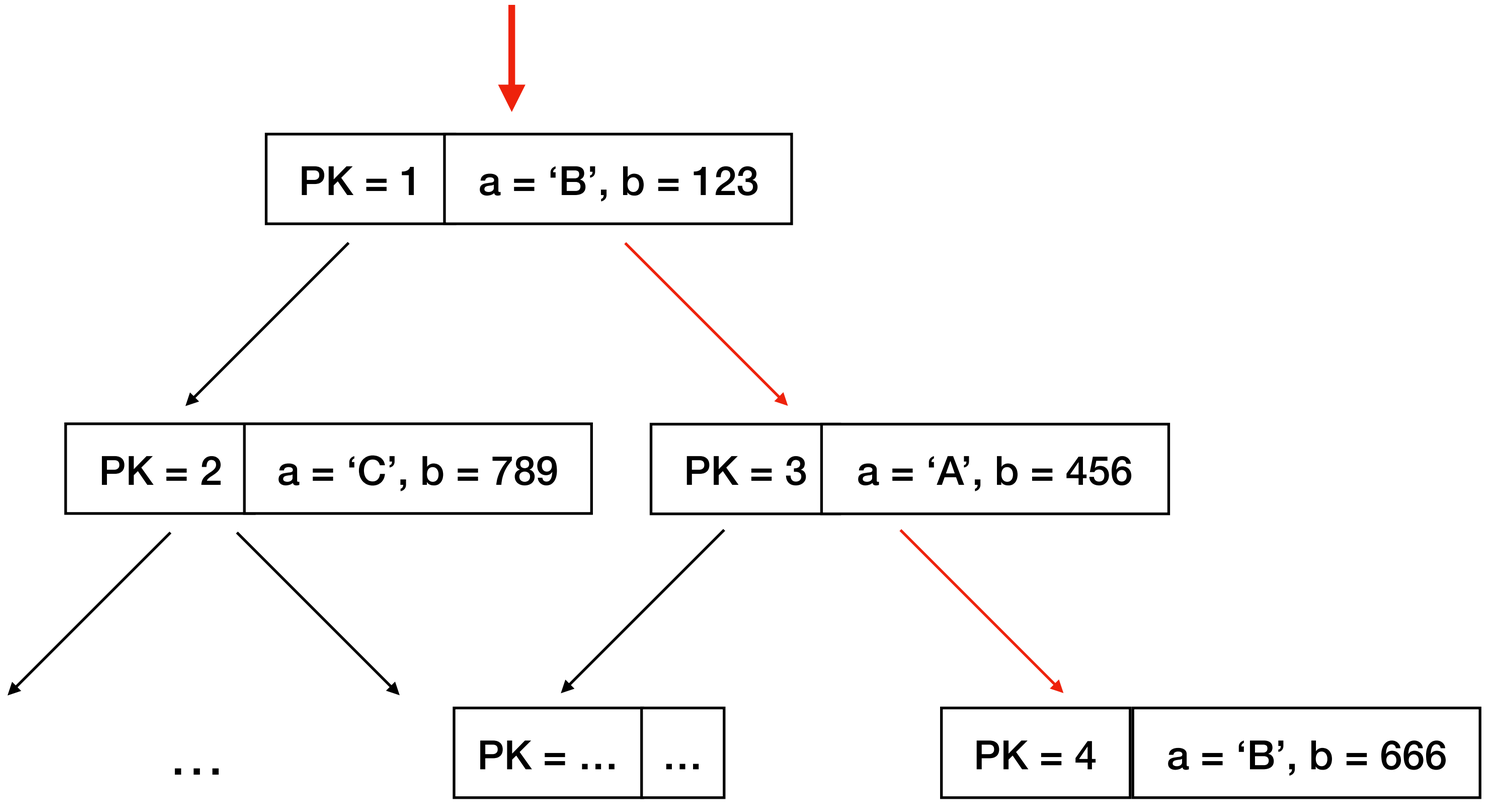
Составной запрос

```
CREATE TABLE t1(id INTEGER PRIMARY KEY, a FLOAT, b VARCHAR, c VARCHAR COLLATE “unicode_ci")
CREATE TABLE t3(id INTEGER PRIMARY KEY, a FLOAT, b VARCHAR, c VARCHAR COLLATE "unicode_ci")

tarantool> box.sql.execute([[SELECT a,b,c FROM t1
                             INTERSECT
                               SELECT a,b,c FROM t1 WHERE b<'d'
                             INTERSECT
                               SELECT a,b,c FROM t3
                             EXCEPT
                               SELECT b,c,a FROM t3
                             ORDER BY c COLLATE "unicode_ci"]])
---
- - [1, 'a', 'a']
  - [9.9, 'b', 'B']
  - [null, 'C', 'c']
...
```

Оптимизатор запросов

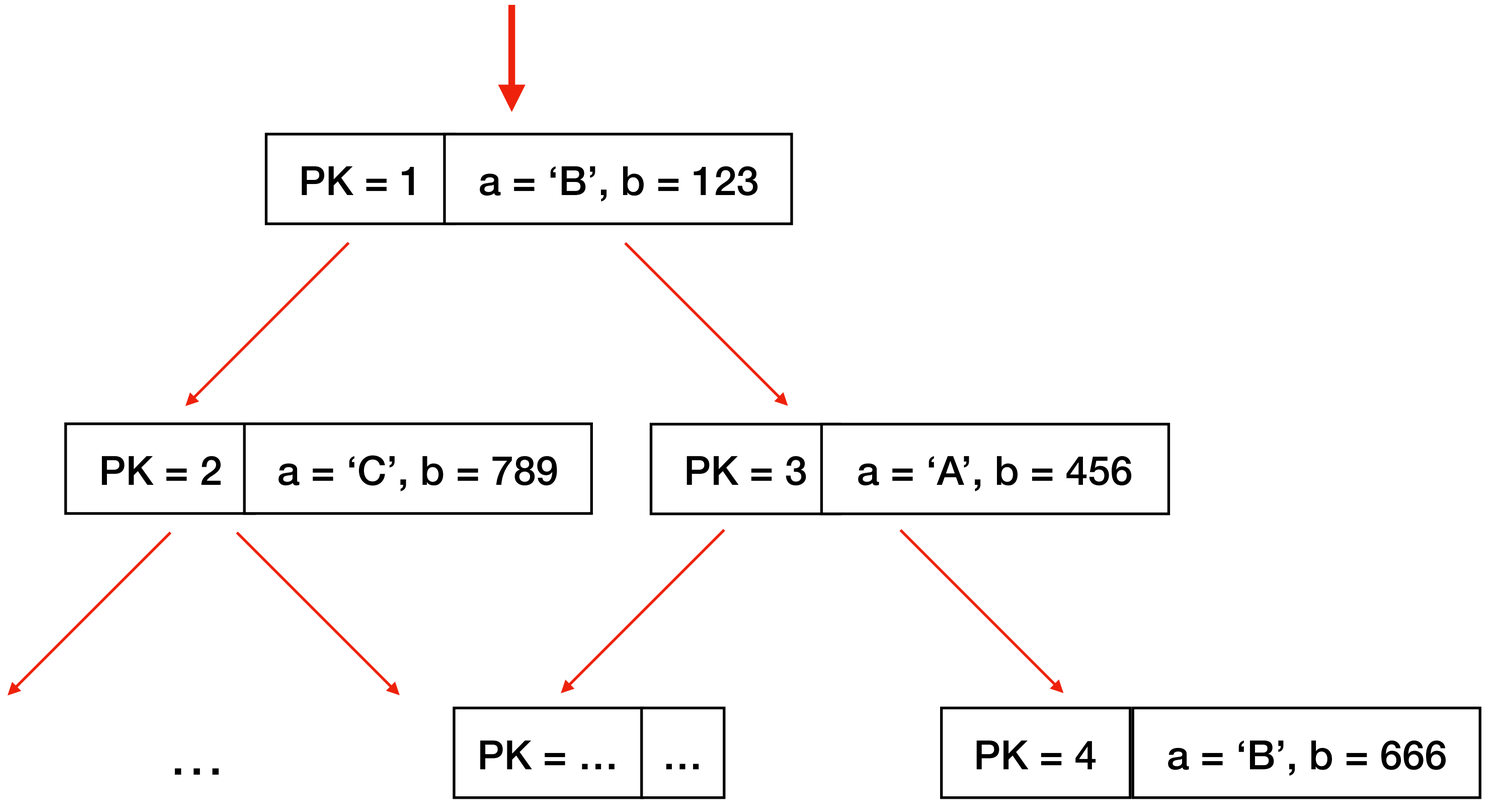
Особенности планировщика Search Table

```
SELECT * WHERE PK = 4;
```

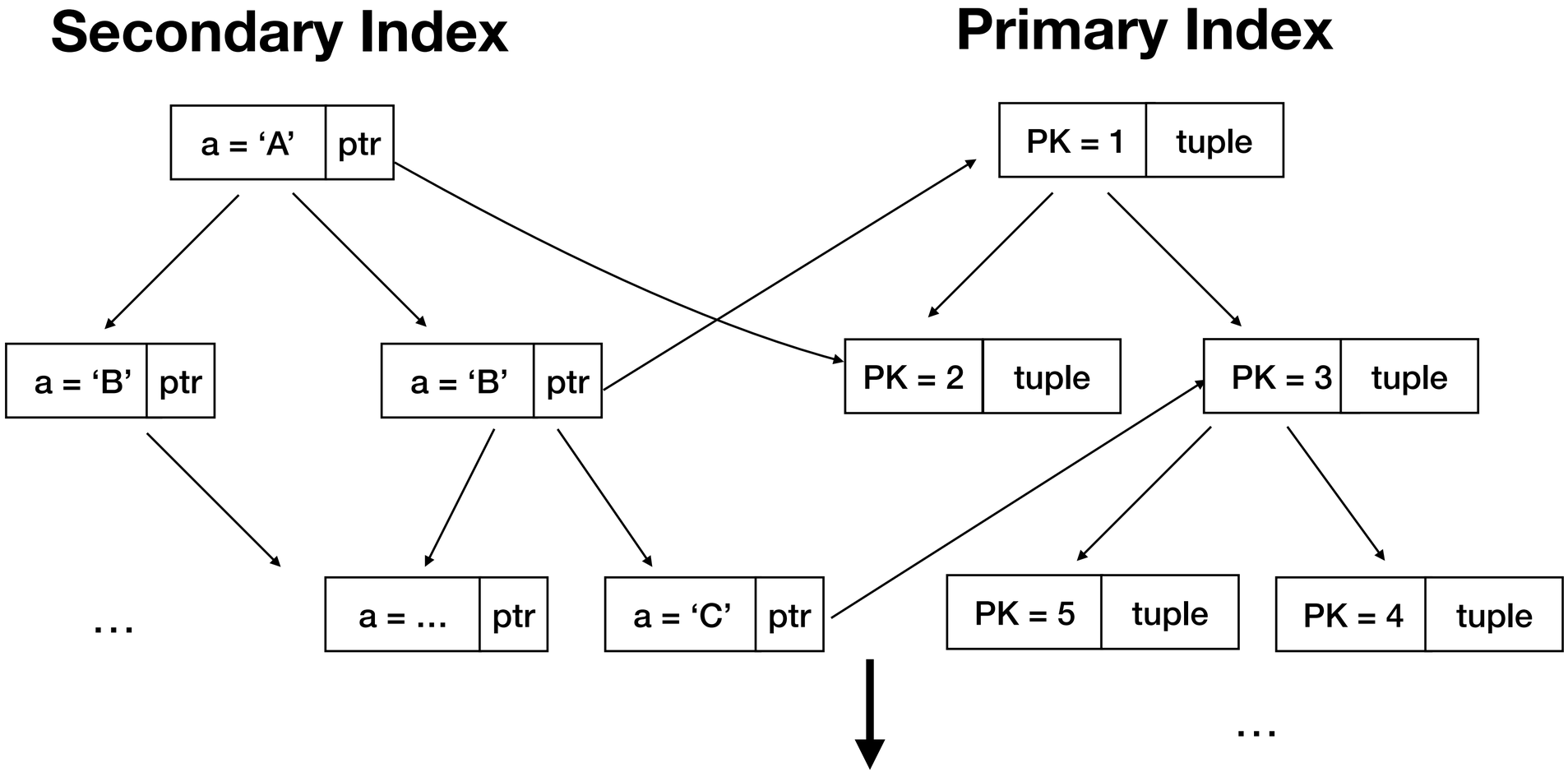


Особенности планировщика Scan Table

```
SELECT * WHERE b = 666;
```



Secondary Indexes



Для memtx все индексы являются покрывающими (COVERING)

Выбор индекса

```
CREATE TABLE t1(id PRIMARY KEY, x, y);

CREATE INDEX t1x ON t1(x);
CREATE INDEX t1y ON t1(y);

SELECT * FROM t1 WHERE x > 0 AND x < 100 AND y > 99;
```


17

Планировщик запросов

1. Query Transformer

   Ex.: (A ⋈ (B ⋈ C)) == (A ⋈ (C ⋈ B))

2. Plan generator

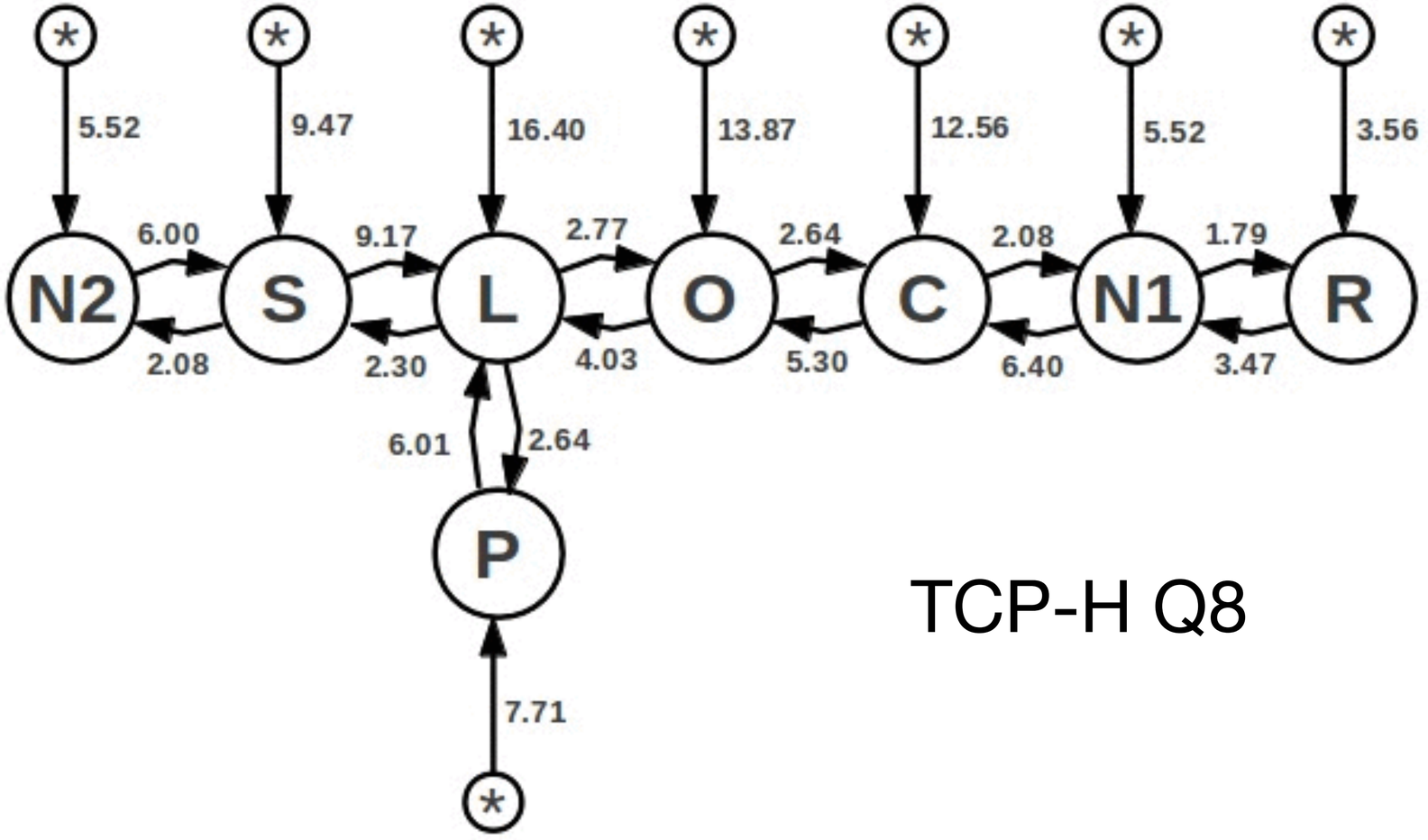
   Ex.: Hash join vs Nested-loop join

3. Plan Estimator

TCP-H Q8

```
SELECT O_YEAR, SUM(CASE WHEN NATION = 'BRAZIL' THEN VOLUME ELSE 0 END)/SUM(VOLUME) AS MKT_SHARE

FROM (SELECT datepart(yy,O_ORDERDATE) AS O_YEAR, L_EXTENDEDPRICE*(1-L_DISCOUNT) AS

                    VOLUME, N2.N_NAME AS NATION FROM PART, SUPPLIER, LINEITEM,

                    ORDERS, CUSTOMER, NATION N1, NATION N2, REGION

 WHERE P_PARTKEY = L_PARTKEY AND S_SUPPKEY = L_SUPPKEY AND L_ORDERKEY = O_ORDERKEY

 AND O_CUSTKEY = C_CUSTKEY AND C_NATIONKEY = N1.N_NATIONKEY AND

 N1.N_REGIONKEY = R_REGIONKEY AND R_NAME = 'AMERICA' AND S_NATIONKEY = N2.N_NATIONKEY

 AND O_ORDERDATE BETWEEN '1995-01-01' AND '1996-12-31' AND P_TYPE= 'ECONOMY ANODIZED STEEL') AS ALL_NATIONS

GROUP BY O_YEAR ORDER BY O_YEAR
```

План запроса: динамическое программирование



TCP-H Q8

Оценка индекса

1. Selectivity
   - Secondary index search: 4 * N
   - PK search: 3 * N
   - Skip-Scan: N * (min(U - L, 1) / N)

2. Startup Cost:
   - ORDER BY: (3.0 * N * log(N)) * (Y/X)
   - Auto-Indexes: X * N * log2(N), X = {7, 1.375}

3. Total Cost

Селективность без статистики

- WHERE x < ? уменьшает кол-во строк в 4 раза
- WHERE x = {0, 1, -1} — в 2 раза
- WHERE x>? AND x<? в 64 раза
- Таблица в среднем содержит около 260000 строк

Выбор индекса

```
CREATE TABLE t1(id PRIMARY KEY, x, y);
```

x, y — равномерное распределение [0, 100]

```
CREATE INDEX t1x ON t1(x);
CREATE INDEX t1y ON t1(y);
```

```
SELECT * FROM t1 WHERE x > 0 AND x < 100 AND y > 99;
```

Статистика

- ANALYZE;
- _sql_stat1(tbl, idx, stat)
- Гистограмма _sql_stat4
- Можно управлять вручную: noskipscan, unordered
- EXPLAIN QUERY PLAN

CBO: Range Analysis

```
CREATE INDEX i1 ON tab(x);
CREATE INDEX i2 ON tab(y);

SELECT z FROM tab WHERE
       x BETWEEN 1 AND 100 AND
       y BETWEEN 1 AND 100;
```

x IN [0, 1000000] —> x10000
y IN [0, 1000] —> x10

CBO: Skip-Scan

```
CREATE INDEX idx ON people(role, age);
SELECT name FROM people WHERE age >= 28;
```

```
SELECT name FROM people WHERE role = 'student' AND age >28
UNION ALL
SELECT name FROM people WHERE role = 'teacher' AND age > 28
```

RBO: Subquery Flattening

```
SELECT a FROM
(SELECT x+y AS a FROM t1 WHERE z<100) WHERE a>5;
```

```
SELECT x+y AS a FROM t1 WHERE z<100 AND a>5;
```

Без flattening’а происходит материализация во временную таблицу

RBO: VIEW Flattening

```
CREATE VIEW v AS SELECT t1.a FROM t1, t2
                        (WHERE t1.a = t2.id OR t1.id = t2.b);

SELECT v.a FROM v, t2 WHERE v.a = t2.b;
```

```
SELECT t1.a FROM t1, t2, t2
        (WHERE t1.a = t2.b AND (t1.a = t2.id OR t1.id = t2.b));
```

Материализации VIEW во временную таблицу НЕТ

RBO: Predicate Push Down

```
SELECT * FROM
(SELECT a AS x, c-d AS y FROM t1)
WHERE x=5 AND y=10;
```

```
SELECT * FROM
(SELECT a AS x, c-d AS y FROM t1 WHERE a=5 AND c-d=10);
```

Co-routines

- Use co-routines to defer computations after sort
- SELECT ALL — Disable Co-Routines

```
SELECT f(a) FROM t ORDER BY date DESC LIMIT 5;
```

```
SELECT f(a) FROM
        (SELECT a FROM t ORDER BY date DESC LIMIT 5);
```


30

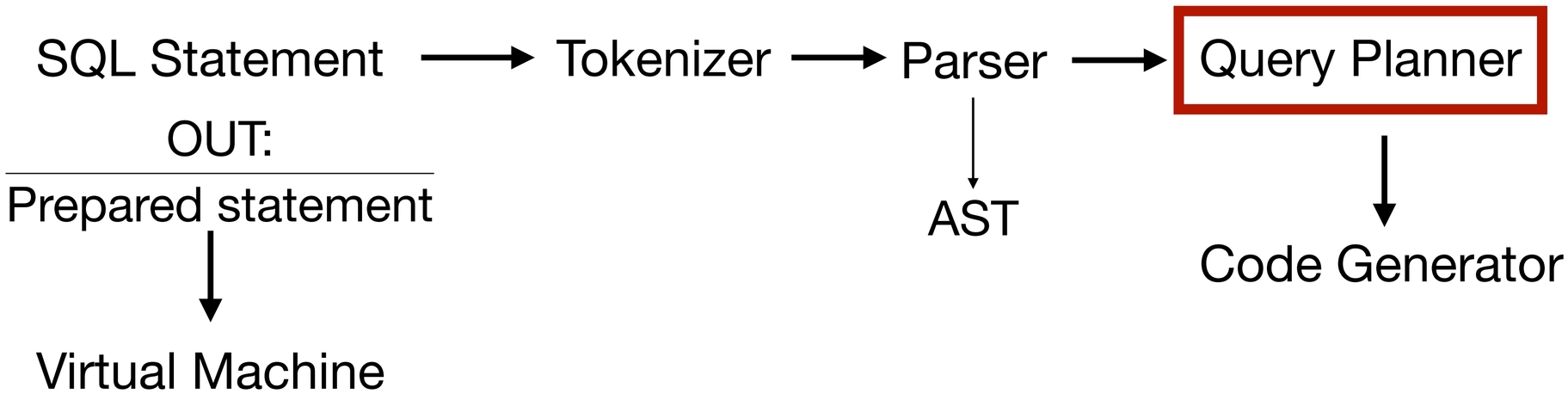
Other RBO

- Xfer: **`INSERT INTO t1 AS SELECT FROM t2;`**
- Simple Select Pattern: 1 table, EQ constraints
- x IN (a, b) —> x = a OR x = b —> Parser transformation
  x = a OR x = b OR x = c —> x IN (a, b, c)
  —> Optimizer transformation
- Simple conjunctions tranformations:

  x<y OR x=y —> x<=y
  x=y OR x=y —> x=y
  x<=y OR x<y —> x<=y

31

Ход выполнения запроса



Пример выполнения

```
SELECT * FROM t1 WHERE PK = 4;
```

- [0, 'Init', 0, 1, 0, '', '00', 'Start at 1']
- [1, 'LoadPtr', 0, 1, 0, 'space<name=T1>']
- [2, 'OpenWrite', 1, 0, 1, '', '02', 'index id = 0, space ptr = 1;]
- [3, 'Explain', 0, 0, 0'']
- [4, 'Integer', 4, 2, 0, '', '00', 'r[2]=4']
- [5, 'SeekGE', 1, 11, 2, '1', '00', 'key=r[2]']
- [6, 'IdxGT', 1, 11, 2, '1', '00', 'key=r[2]']
- [7, 'Column', 1, 0, 3, '', '00', 'r[3]=T1.PK']
- [8, 'Column', 1, 1, 4, '', '00', 'r[4]=T1.A']
- [9, 'Column', 1, 2, 5, '', '00', 'r[5]=T1.B']
- [10, 'ResultRow', 3, 3, 0, '', '00', 'output=r[3..5]']
- [11, 'Halt', 0, 0, 0, '', '00', '']

Инициализация и завершение

```
SELECT * FROM t1 WHERE PK = 4;
```

```
- [0, 'Init', 0, 1, 0, '', '00', 'Start at 1']
- [1, 'LoadPtr', 0, 1, 0, 'space<name=T1>']
- [2, 'OpenWrite', 1, 0, 1, '', '02', 'index id = 0, space ptr = 1; ]
- [3, 'Explain', 0, 0, 0'']
- [4, 'Integer', 4, 2, 0, '', '00', 'r[2]=4']
- [5, 'SeekGE', 1, 11, 2, '1', '00', 'key=r[2]']
- [6, 'IdxGT', 1, 11, 2, '1', '00', 'key=r[2]']
- [7, 'Column', 1, 0, 3, '', '00', 'r[3]=T1.PK']
- [8, 'Column', 1, 1, 4, '', '00', 'r[4]=T1.A']
- [9, 'Column', 1, 2, 5, '', '00', 'r[5]=T1.B']
- [10, 'ResultRow', 3, 3, 0, '', '00', 'output=r[3..5]']
- [11, 'Halt', 0, 0, 0, '', '00', '']
```

Создание курсора

```
SELECT * FROM t1 WHERE PK = 4;
```

```
- [0, 'Init', 0, 1, 0, '', '00', 'Start at 1']
- [1, 'LoadPtr', 0, 1, 0, 'space<name=T1>']
- [2, 'OpenWrite', 1, 0, 1, '', '02', 'index id = 0, space ptr = 1; ]
- [3, 'Explain', 0, 0, 0'']
- [4, 'Integer', 4, 2, 0, '', '00', 'r[2]=4']
- [5, 'SeekGE', 1, 11, 2, '1', '00', 'key=r[2]']
- [6, 'IdxGT', 1, 11, 2, '1', '00', 'key=r[2]']
- [7, 'Column', 1, 0, 3, '', '00', 'r[3]=T1.PK']
- [8, 'Column', 1, 1, 4, '', '00', 'r[4]=T1.A']
- [9, 'Column', 1, 2, 5, '', '00', 'r[5]=T1.B']
- [10, 'ResultRow', 3, 3, 0, '', '00', 'output=r[3..5]']
- [11, 'Halt', 0, 0, 0, '', '00', '']
```

Позиционирование курсора

```
SELECT * FROM t1 WHERE PK = 4;
```

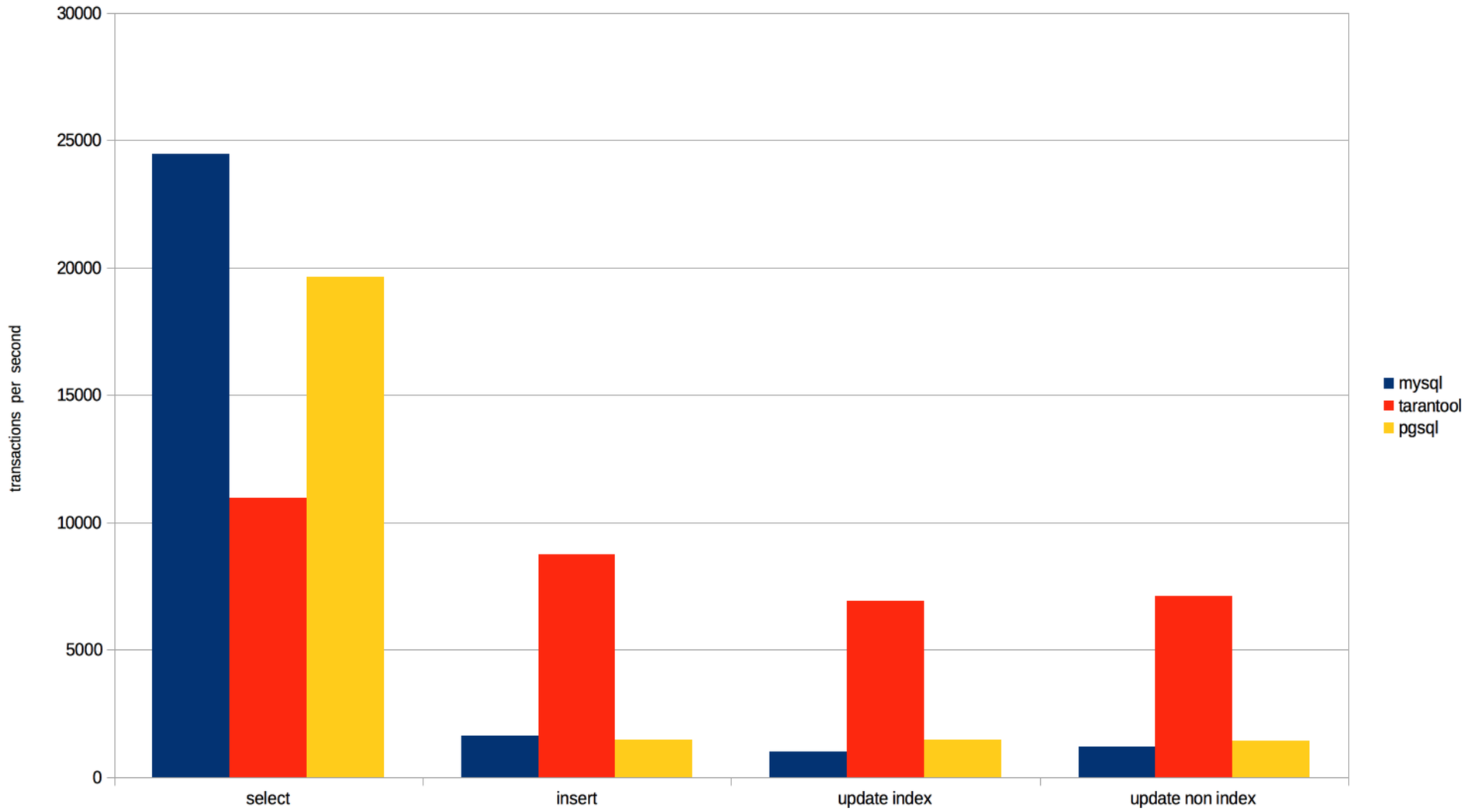
- [0, 'Init', 0, 1, 0, '', '00', 'Start at 1']
- [1, 'LoadPtr', 0, 1, 0, 'space<name=T1>']
- [2, 'OpenWrite', 1, 0, 1, '', '02', 'index id = 0, space ptr = 1;]
- [3, 'Explain', 0, 0, 0'']
- [4, 'Integer', 4, 2, 0, '', '00', 'r[2]=4']
- [5, 'SeekGE', 1, 11, 2, '1', '00', 'key=r[2]']
- [6, 'IdxGT', 1, 11, 2, '1', '00', 'key=r[2]']
- [7, 'Column', 1, 0, 3, '', '00', 'r[3]=T1.PK']
- [8, 'Column', 1, 1, 4, '', '00', 'r[4]=T1.A']
- [9, 'Column', 1, 2, 5, '', '00', 'r[5]=T1.B']
- [10, 'ResultRow', 3, 3, 0, '', '00', 'output=r[3..5]']
- [11, 'Halt', 0, 0, 0, '', '00', '']

Декодирование MsgPack

```
SELECT * FROM t1 WHERE PK = 4;
```

```
- [0, 'Init', 0, 1, 0, '', '00', 'Start at 1']
- [1, 'LoadPtr', 0, 1, 0, 'space<name=T1>']
- [2, 'OpenWrite', 1, 0, 1, '', '02', 'index id = 0, space ptr = 1; ]
- [3, 'Explain', 0, 0, 0'']
- [4, 'Integer', 4, 2, 0, '', '00', 'r[2]=4']
- [5, 'SeekGE', 1, 11, 2, '1', '00', 'key=r[2]']
- [6, 'IdxGT', 1, 11, 2, '1', '00', 'key=r[2]']
- [7, 'Column', 1, 0, 3, '', '00', 'r[3]=T1.PK']
- [8, 'Column', 1, 1, 4, '', '00', 'r[4]=T1.A']
- [9, 'Column', 1, 2, 5, '', '00', 'r[5]=T1.B']
- [10, 'ResultRow', 3, 3, 0, '', '00', 'output=r[3..5]']
- [11, 'Halt', 0, 0, 0, '', '00', '']
```

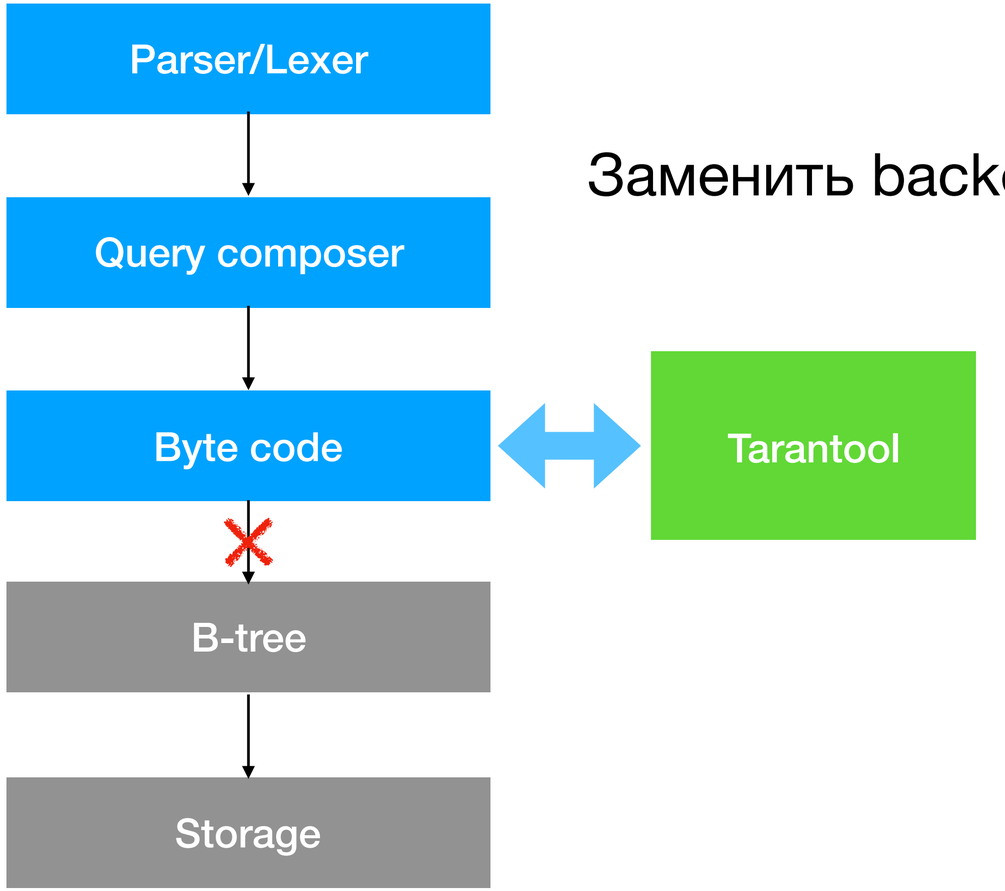
Sysbench, 1 core, 1GB



Использование SQL

- Из Lua: box.sql.execute(“select * from t”)
- \set language sql
- Tarantool-C: chunked results

Реализация



Заменить backend SQLite

Реализация-1

- Самодельный лексер
- Легковесный парсер (400 состояний, Lemon)
- Stmt-based virtual machine (VDBE)
- Выделенный storage engine API (B-tree)
- 120 kSlocs
- Быстрая
- Относительно просто устроена
- Легко декомпонируется

Чего пока нет

- Курсоры
- Привилегии
- Information schema
- SQL/PSM
- DATETIME, DECIMAL

- MEMTX engine
- B-tree indexes
- SCALAR type
- One directional iterators
- BOX interface overhead
- Lack of partial and functional indexes

Статус

- Тестовое покрытие сохранено
- 95% pass rate
- Состояние - alpha-1 (2.0)
- Beta: Mid 2018

Ultima Thule

- JIT
- Cluster
- Multiple SQL dialects
- Compiled query offloading/serialization

Спасибо

korablev@tarantool.org

kyukhin@tarantool.org